Lab 8
Modified Instructions

EELE 467
Due date: 11/05/2024

The following sections highlight any changes or additions compared to the instructions in the textbook. Please read the textbook instructions first, and then read these instructions carefully.

You will be expected to follow the C style guide located at `docs/style-guides/c-style.md` in your repository!

If you prefer, you can write your code on Windows and compile/test it in WSL. This is more lightweight than running your VM. Regardless, once you start accessing memory-mapped I/O (i.e., reading/writing your registers), you will need to run your code on the SoC.

Please use getopt or some other argument parsing library for parsing your arguments. It will make your life easier in the long run.

Program requirements

C program requirements, Requirement 1 Please name your program `led-patterns.c` or something similar. Please don't put "my" at the beginning of any of your filenames...Be consistent with the naming conventions we have been using thus far: use hyphens or underscores in file/folder names.

C program requirements, Requirement 5 The program should print the help/usage text when the program is run with no arguments or incorrect arguments.

C program requirements, Requirement 6 When terminating your program with ctrl+c, your program must set your hardware back into "hardware control mode" (i.e., set `hps_led_control` back to `0`).

Code location Put your source code in `sw/led-patterns`.

Usage text Please format your usage text similar to how most Unix tools do. Look at the help output of common commands like `ls`, etc. You may wish to follow the usage-text conventions of Python's argparse module (IMO, the format is quite nice); here's an example:

```
usage: myprogram [-h] [-f FOO]

options:
  -h       show this help message and exit
  -f FOO   foo of the myprogram program
```

Lab steps

Lab steps, Step 1 In Step 1 of the lab, you are told to create an rbf file so your FPGA can be programmed upon boot. If you have not already done this for Lab 7, please do this. However, rather than copying the rbf file to `/srv/tftp/...` like the lab says, you will copy it to the FAT32 partition of your SD card.

Lab steps, Step 4: cross compiling You will need to compile your program with the `-static` flag as you did in Lab 7. We will update the SoC's root filesystem soon so we don't have to use static linking...

Lab steps, Step 4: program location Since we have not set up an NFS server yet, you will copy your compiled program to the EXT4 partition of your SD card. This has to be done in your Ubuntu VM since Windows can't read the EXT4 partition.

Demonstration

Create two different patterns text files and demonstrate both of them. You still need to do the demos listed in the textbook; you will simply have two text files instead of one.

Deliverables

GitHub Submission

Follow the workflow given in your repository (`docs/workflow.md`), per usual. Make sure all your files are committed.

LED patterns README

Create a `README.md` file in `sw/led-patterns` that describes your program and it's usage. Your README should include

- a "Usage" section that displays your program's usage text and some usage examples.
- a "Building" (or "Compilng" or "Installation") section that describes how to compile your code.
- some documentation explaining that your code requires the FPGA to be programmed with your LED patterns bitstream; this could be in it's own section or part of the "Usage" or "Building" sections.

`docs/lab8.md`

Somewhere in your `lab8.md` file, create a link to your `sw/led-patterns/README.md` file.

Create a section in your `lab8.md` file describing how your calculated the physical addresses of your component's registers.